

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EKSAMEN/TOETS**  **EXAMINATION/TEST:** | **Semester test 2017 MEMO** | **KWALIFIKASIE/**  **QUALIFICATION:** | **BSc** | |
| **MODULEKODE/**  **MODULE CODE:** | **ITRW222** | | **TYDSDUUR/**  **DURATION:** | **2 hours** |
| **MODULEBESKRYWING/**  **MODULE DESCRIPTION:** | **Datastrukture/**  **Data Structures** | | **MAKS/**  **MAX:** | **60** |
| **EKSAMINATOR(E)/**  **EXAMINER(S):** | **Prof. R Goede** | | **DATUM/**  **DATE:** | **26/09/17** |
| **MODERATOR/**  **MODERATOR:** |  | | **TYD/TIME:** | **10:00** |

**Vraag 1/ *Question 1* (25)**

|  |  |
| --- | --- |
| Gebruik die *τ -* notasie om die looptyd van die volgende programlyne te bepaal. (15) | *1.1 Use* ***τ******-*** *notation to determine* ***the running time*** *of the following program lines (15)* |
| 1. for ( int i=0; i<=n; i++) {  2. b=arr[i]+1; } | |
| 1a. tfetch √+ tstore √  1b. (2tfetch √+ t< √)(n+2) √  1c. (2tfetch √+ t+ √+ tstore √)(n+1) √  2. (4tfetch √√+ t+ √+ tstore √+ t[.])√ (n+1) √ | |

|  |  |  |
| --- | --- | --- |
| 1. 1.2 Bepaal die looptyd van al drie dele van lyn 8 in konteks van hierdie programdeel. Jy hoef nie die uitdrukkings te vereenvoudig nie. Maak gebruik van die vereenvoudigde model. (8) | | *1.2 . Determine the running time of all three parts of line 8 in context of this program segment. You need not simplify the expressions. Use the simplified model. (8)* |
| 1 public class Question1.2  2 {  3 public static int numbers (int n)   1. { 2. int prod = 1; 3. for (int i=1; i<n; i++ ) 4. { 5. for ( int j=0; j<i; ++j) 6. prod \*=j; 7. } 8. return prod; 9. } 10. } | | |
| 8a 2(n-1)√ √  8b √√√  8c √√√ | | |
| 1.3 Gee die definisie van O(n) (2) | *1.3.* Give the definition for: O(n) (2) | |
| Consider a function f(n) that is non-negative for all inters n>=0. We say that “f(n) is big oh g(n)” which we write f(n) = O(g(n)), if there exists an inter n0 and a constant c> 0 so that for all integers n>=n0, f(n) <= c g(n). √√ | | |

**Vraag 2 / *Question 2* (5)**

|  |  |
| --- | --- |
| 2. Aanvaar die volgende kode bestaan: Ontwerp ’n metode vir die klas Stack genaamd: *push()* in Java (5) | 2. Assume the following code exists.  Design a method for the class Stack called: *push()* in Java. (5) |
| public class Stack  { private Listing[] data;  private int top;  private int size;  public Stack( )  { top = -1;  size = 100;  data = new Listing[100];  } | |
| Ontwerp ’n metode vir die klas Stack genaamd: *push()*  public boolean push(Listing newNode) √  { if(top == size-1) √  return false; // \*\* overflow error \*\*√  else  { top = top +1; √  data[top] = newNode.deepCopy();√  return true; // push operation successful  }  } | |

**Vraag 3/ *Question 3* (25)**

|  |  |
| --- | --- |
| 3 Bestudeer die volgende klas: | *3 Study the following class:* |
| public class SLL<T extends Comparable<? super T>>  {  private Element<T> head; // list header  private Element<T> tail;    public SLL()  { head = null;  tail = null;}  …  public class Element<T1 extends Comparable<? super T>>  {  private T1 data;  private Element<T1> next;  public Element(T1 param)  {  data = param;  }  }// end of inner class Node  } | |
| 3.1 Skryf ‘n metode in Java vir die klas SLL genaamd append() wat ‘n nuwe element aan die einde van die lys byvoeg. (5) | *3.1 Write a method in Java for the class SLL called append that adds a new element at the back of the list (5)* |
| *public boolean append(T newElement) //insert at tail*  *{*  *Element<T> temp = new Element<T>(newElement);*  *if(temp == null) // out of memory*  *return false;*  *else*  *{*  *if (head==null)*  *{*  *head = temp;*  *tail = temp;*  *}*  *else*  *{*  *tail.next = temp;*  *tail = temp;*  *}*  *}*  *return true;*  *}* | |
| 3.2 Ontwerp ‘n metode genaamd:*filter( …)* om sekere elemente uit ‘n geskakelde lys te verwyder. Jou metodoe ontvang ‘n GESKAKELDE LYS wat PRESIES uit TWEE element bestaan (jy moet dit toets) as parameter. Filter() moet al die elemente uit die roepende lys uit haal wat KLEINER as die eerste PARAMETER LYS item is en wat GROTER AS die tweede PARAMETER LYS item is: bv. *Roepende lys = {3,1,5,8,6,5,4,3,7,1,8}*  *parameter lys= {4,7}*  *NA filter:*  *Roepende lys = {5,6,5,4,7}*  Jy moet die kode gee vir al die bestaande SLL metodes wat jy wil gebruik, behalwe toString() | *3.2 Design a method for the class called:*  **filter(…)** *to remove certain elements from the list. Your method receives a LINKED LIST containing PRECISELY TWO element (you need to verify this) as parameter. Filter() should remove all the elements in the calling list which are SMALLER than the first PARAMETER LIST item and LARGER THAN the second PARAMETER LIST item: eg.*  *calling List= {3,1,5,8,6,5,4,3,7,1,8}*  *parameter list= {4,7}*  *AFTER filter:*  *Calling List = {5,6,5,4,7}*  You have to give all the code for existing SLL methods you want to use, except toString() |
| 3.2.1 Teken ‘n geskakelde lys om jou met die algotirme-ontwerp te help.  Skryf die algemene en die spesiale gevalle vir die probleem in Afrikaans neer. Gee ‘n kort beskrywing van die nodige aksie vir elkeen van die gevalle. (5) | *3.2.1 Draw a linked list to help you to design the algorithm.*  *Write down the general and all the special cases for the problem in English. Give a short description of the required action for each of these cases. (5)* |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | *Calling list* |  | *A200* |  | *B145* |  | *C001* |  | *DDDD* | | *Head = A200* |  | *5* |  | *7* |  | *9* |  | *12* | | *Tail = DDD* |  | *B145* |  | *C001* |  | *DDDD* |  | *null* |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | *Param list* |  | *A222* |  | *BB11* |  |  | | *Head = A222* |  | *6* |  | *7* |  |  | | *Tail = B111* |  | *BB11* |  | *null* |  |  |   *General – traverse calling list with while (Ptr!= null) and test each element at ptr – if it is larger than paramList.head and smaller than paramList.tail – add to new list*  *Change new list to calling list by copying head and tail*  *Special*  *Neither lists should be empty- so test if any one is empty – return without any action*  *Parameter list should have exactly 2 elements test that head.next== tail* | |
| 3.2.2. Skryf die metode ***filter(…)*** in Java. (12) | *3.2.2 Write the method* ***filter(…)*** *in Java*. (12) |
| *public void filter(SLL<T> paramList)*  *{*  *Element<T> paramPtr; // declare variable paramPtr to traverse parameter list*  *Element<T> thisPtr; // declare variable thisPtr to traverse calling list*  *//create new list for output*  *SLL<T> filteredList = new SLL<T>();*    *if (paramList.head==null|| head==null) // any of the lists empty?*  *{*  *System.out.println("\nEmpty list");*  *return;*  *}*  *// test if paramlist has 2 elements*  *if (paramList.head.next != paramList.tail)*  *{*  *System.out.println("\nParameter list should have 2 elements");*  *return;*  *}*    *// traverse calling list and add correct values to new list*    *thisPtr = head; // start at beginning of calling list*    *while (thisPtr != null) // continue until end of calling list*  *{*  *if (( thisPtr.data.compareTo(paramList.head.data)>=0) &&*  *(thisPtr.data.compareTo(paramList.tail.data)<=0))*  *// if thisPtr.data>=paramList.head.data AND <= paramList.tail.data*  *{*  *filteredList.append(thisPtr.data);*  *}*  *thisPtr=thisPtr.next;*  *}*  *// copy filteredList to calling list;*  *head=filteredList.head;*  *tail = filteredList.tail;*  *}* | |
| 3.2.3 Skryf ‘n drywer program om die metode deeglik te toets. (8) | 3.2.3 W*rite a driver program to test the method thoroughly. (8)* |
| public class Driver  {  public static void main(String [] args)  {    SLL<Integer> myList = new SLL<Integer>();  SLL<Integer> filterValues = new SLL<Integer>();      // test filter with empty list  myList.filter(filterValues);  System.out.printf("\n ", myList);  System.out.println("\nAdd items: ");// Add item to Empty list  myList.append(new Integer(2));  myList.append(new Integer(1));  myList.append(new Integer(6));  myList.append(new Integer(1));  myList.append(new Integer(2));  myList.append(new Integer(7));  myList.append(new Integer(6));  myList.append(new Integer(11));  System.out.printf("\nmylist %s",myList);  System.out.printf("\nfilter %s",filterValues);  // test filter with calling list and empty filter list  myList.filter(filterValues);  System.out.printf("\n %s", myList);  //test with too short parameter  filterValues.append(new Integer(4));  myList.filter(filterValues);  System.out.printf("\n %s", myList);  // test general case  filterValues.append(new Integer(7));  myList.filter(filterValues);  System.out.printf("\nMylist %s",myList);  // Test with too many filter values  filterValues.append(new Integer(7));  myList.filter(filterValues);  System.out.printf("\nMylist %s",myList);  }  } | |

**Vraag 4/ *Question 4* (5)**

|  |  |
| --- | --- |
| 4.1 Verduidelik in jou eie woorde hoe ‘n “hash” table data stoor (3) | 4.1 Explain in your own words how a hash table stores data. (3) |
| When a mapping is done between the KEY FIELD VALUE and storage index TO PREVENT SEQUENCIAL search of an array = A formula is used to map the key field to the memory address | |
| 4.2 Wat is ‘n perfekte “hashing” – funksie? (2) | 4.2 What is a perfect hashing function? (2) |
| When the hashing algorithm returns a unique value for the address. | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Marking scheme Question 3** | |  |  |
| **Question 1** |  |  |  |
| Heading | 0 | 1 |  |
| Create element | 0 | 1 |  |
| Test for empty list | 0 | 1 |  |
| update head and tail if empty | 0 | 1 |  |
| Update tail if not empty | 0 | 1 |  |
|  |  | **5** |  |
| **Question 2** |  |  |  |
| Diagram – calling list | 0 | 1 | 2 |
| Diagram – parameter list | 0 |  | 1 |
| General case description | 0 |  | 1 |
| Special case description | 0 |  | 1 |
|  |  |  |  |
| Heading of method | 0 | 1 |  |
| Create object of SLL for third list | 0 | 1 |  |
| test if any lists are empty -empty list | 0 | 1 |  |
| Test if param has 2 elements | 0 | 1 |  |
| Declare ptr == head | 0 | 1 |  |
| Traverse list | 0 | 1 |  |
| If ptr.info > == head | 0 | 1 |  |
| If ptr.info < == tail | 0 | 1 |  |
| Correct use of compareTo() | 0 | 1 |  |
| Add correct numbers to list 3 | 0 | 1 |  |
| Copy list 3 to calling list head | 0 | 1 |  |
| Copy list 3 to calling list tail | 0 | 1 |  |
|  |  | **12** |  |
| **Question 3** |  |  |  |
| create about 5 objects of Integer class | 0 | 1 |  |
| Test empty list for both list | 0 | 1 |  |
| Add elements into both lists | 0 | 1 |  |
| Test general case | 0 | 1 |  |
| Test too short param list | 0 | 1 |  |
| Test too long param list | 0 | 1 |  |
| Appropriate printing methods | 0 | 2 |  |
| **-2 if numbers are added to lists rather than objects** |  | **8** |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| **TOTAL** |  | 25 |  |